



## Strathprints Institutional Repository

Dunlop, Mark and Levine, John (2012) *Multidimensional Pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking*. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, New York, pp. 2669-2678. ISBN 978-1-4503-1015-4

Strathprints is designed to allow users to access the research output of the University of Strathclyde. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. You may not engage in further distribution of the material for any profitmaking activities or any commercial gain. You may freely distribute both the url (<http://strathprints.strath.ac.uk/>) and the content of this paper for research or study, educational, or not-for-profit purposes without prior permission or charge.

Any correspondence concerning this service should be sent to Strathprints administrator: <mailto:strathprints@strath.ac.uk>

# Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed, Familiarity and Improved Spell Checking

Mark D Dunlop and John Levine

Computer and Information Sciences, University of Strathclyde  
Richmond Street, Glasgow, G1 1XH, Scotland, UK  
mark.dunlop@cis.strath.ac.uk, john.levine@cis.strath.ac.uk

## ABSTRACT

This paper presents a new optimization technique for keyboard layouts based on Pareto front optimization. We used this multifactorial technique to create two new touchscreen phone keyboard layouts based on three design metrics: minimizing finger travel distance in order to maximize text entry speed, a new metric to maximize the quality of spell correction by reducing tap ambiguity, and maximizing familiarity through a similarity function with the standard Qwerty layout. The paper describes the optimization process and resulting layouts for a standard trapezoid shaped keyboard and a more rectangular layout. Fitts' law modelling shows a predicted 11% improvement in entry speed without taking into account the significantly improved error correction potential and the subsequent effect on speed. In initial user tests typing speed dropped from approx. 21 wpm with Qwerty to 13 wpm (64%) on first use of our layout but recovered to 18 wpm (85%) within four short trial sessions, and was still improving. NASA TLX forms showed no significant difference on load between Qwerty and our new layout use in the fourth session. Together we believe this shows the new layouts are faster and can be quickly adopted by users.

## Author Keywords

Touch-screen; keyboard design; keyboard optimization

## ACM Classification Keywords

H.5.2 User Interfaces: Input devices and strategies

## INTRODUCTION

Text entry on mobile phones has always been a compromise between the space allocated to text entry and the size of the device. With finger-controlled touch screens becoming dominant in the late 00's this problem was exaggerated by the lack of precision when using relatively large blunt fingertips to tap small on-screen buttons and the lack of tactile feedback from touch screens (e.g. [13]). This combination led to higher error rates on touch screen

phones than on physical keyboards [1] and many users using landscape mode to gain larger keyboards at the expense of application display space.

The Qwerty layout has been adopted almost universally on laptops and desktops despite the design constraints being far removed from the early physical typewriters that inspired the layout. Alternatives such as the Dvorak Simplified Keyboard have not been successful for many reasons [5], but largely because of the high initial learning curve when moving from Qwerty to a faster but alien layout. While there have been several faster optimized keyboard layouts for touch screens (e.g. The Opti [24], Metropolis [31] and matrix [20] keyboards), these suffer the same alienation problem as the Dvorak layout. The Qwerty keyboard has, thus, dominated on touch screen phones as pick-up-and-use usability issues have prevented the adoption of more optimal keyboards.

Bi, Smith and Zhai [2] introduced a novel approach to keyboard optimization to attempt to overcome the initial hostility of users to alternative layouts. They allowed the keys of a Qwerty layout to shuffle by at most one position from their original location to achieve a *quasi-optimized Qwerty* variant. This layout had typing speed performance between the original Qwerty layout and a fully-optimized layout while not being alien as keys were roughly where the user would expect them to be. Touch screens and finger interaction users normally focus on the keyboard area during text entry, thus moving keys slightly is less of a problem than one might expect from desktop/laptop physical keyboard use.

With modern powerful touch screen phones has come increasingly powerful error correction. Error correction



Figure 1: Triple optimized rectangular keyboard

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI'12, May 5–10, 2012, Austin, Texas, USA.

Copyright 2012 ACM 978-1-4503-1015-4/12/05...\$10.00.

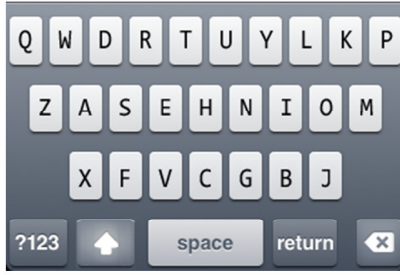


Figure 2: Bi, Smith and Zhai's Quasi-Qwerty layout

methods attempt to correct both users' spelling mistakes and their typing errors – most commonly hitting neighbouring keys to the intended ones (e.g. [18]). Spell checking is made considerably harder when correcting typed words that are, themselves valid even if the context is wrong (e.g. [15]). As an example, the Qwerty layout has the *I* and *O* keys as neighbours, thus *in/on*, *if/of*, *for/fir*, *hot/hit* etc. are all only one key slip from each other. With smaller touch-screen phones this can be a very short physical distance, e.g. on an HTC Hero, key centres are under 4.5 mm apart. The arrangement of the characters on the keyboard can improve the performance of an error correction algorithm by, for a given language, reducing the likelihood of near-misses resulting in valid words. While it has been shown that the layout of ambiguous keyboards, for example the traditional phone 12-key pad, can considerably affect entry performance [10], we believe this paper presents the first work to adjust the layout of an unambiguous keyboard for spell correction.

In the remainder of this paper we present a triple-optimization process using Pareto front optimization that attempts to optimize for (a) speed of text entry, (b) error correction tap interpretation clarity and (c) familiarity to the traditional Qwerty layout. Initially we present the three metrics in detail then their combination through Pareto front optimization. We also present keyboard layouts generated by this process for the traditional key layout and for a slightly squarer layout that increases key sizes (figure 1). Finally, we present results from Fitts' law analysis and an initial study into pick-up-and-use usability of our optimized layout. Throughout the paper we will focus on portrait mode text entry – the normal style of interaction with a touch-screen phone and the larger challenge for text entry.

## OPTIMISATION METRICS

### Finger distance metric

The time taken to type a letter on the keyboard is dependent on two factors: how long it takes the user to move his/her finger to a position above the key and how long it takes to tap the key. Fitts' law [9] has been used extensively to predict the time taken by users to select spatial targets. For design, Fitts' law implies that the nearer and bigger a target is the quicker it is to tap. Fitts' law has been used to model text entry on, for example, traditional phone keypads (e.g. [27]) and stylus based keyboards (e.g. [25]) in attempts to

both predict likely expert performance rates and to design faster keyboard layouts. Fitts' law calculates the time for a single key tap as:

$$T = a + b \log_2 \left( \frac{D}{W} + 1 \right) \quad (1)$$

where  $D$  is the distance to the target key from the starting position and  $W$  is the width of the target key (the constants  $a$  and  $b$  are dependent on the physical characteristics of the keyboard and need to be determined empirically).

Here we constrain the optimization process in two ways:

- We fix the keyboard layout at the start of the optimization procedure: we restrict ourselves to different letter-to-key assignments and not the more general keyboard layout problem of adjusting the button sizes and positions;
- We model single finger text entry: most users of touch-screen phones use the index finger of their dominant hand as the pointer – particularly for small keys [1].

Given these constraints we can simplify from Fitts' law by only modelling the distance that the user's finger has to move to enter text. For comparing two keyboards this is a faster and simpler calculation that is as effective at stating if one keyboard is faster than the other, but without giving full predictions of typing speed. In the optimization process, all keys were modelled as the same size bar the space key which, for simplicity, we modelled as three standard sized keys beside each other on the bottom row – distances were measured to the nearest of the keys (a similar approach to [24] but with a shorter spacebar typical of mobiles).

We built a bigram weighting model of English by using the same national newspaper corpus of English text as in our previous studies [7] (with 77 317 unique words and a total of 5 171 840 occurrences). While the corpus is journalistic in nature, it has been argued that the source of the corpus is not critical to keyboard optimization [30] and our bigrams are similar to previous published ones (e.g. [28]). Our analysis calculated an occurrence count for each two-letter bigram as used in the corpus<sup>1</sup>. To include movement to and from the space key we also included space to give 27\*27 possible letter combinations from the 26 letter alphabet. The most common letter pair was  $E\_$  (where  $\_$  represents space) with 981 920 occurrences in our collection. The probability of any key sequence being  $E\_$  is thus 0.033. The top key combination probabilities<sup>2</sup> are  $E\_$ =0.033,  $\_ T$ =0.026,  $S\_$ =0.022,  $TH$ =0.021,  $HE$ =0.020,  $\_ A$ =0.019. The lowest non-zero pairing was  $ZX$ =0.000 (1 occurrence).

We calculated the weighted average finger distance by summing the product of the Euclidian distance between letters pairs and their relative probability from the corpus:

$$M_{dist} = \langle \sum_{i,j \in \alpha} p_{i,j} \cdot distance(k_i, k_j) \rangle$$

<sup>1</sup> We adjusted the text to include US and UK variants of common words

<sup>2</sup> Full list at <http://personal.cis.strath.ac.uk/~mdd/research/chi2012/>

where  $a$  is the alphabet in use (here  $a...z$  plus *space*),  $p_{ij}$  is the probability of the transition from letter  $i$  to  $j$  in the corpus,  $k_i$  is the key for letter  $i$ , *distance* is the Euclidian distance between the keys' centres.

To evenly balance the multiple criteria optimization process used later in this paper, it is helpful if the metrics have roughly equal ranges of values. We normalized the scores for finger distance to the range of approximately 0...1, where 1 represents the best keyboard found and 0 the worst. We initially derived a fast keyboard iteratively with several short runs of the optimizer. The normalised score was given as  $M_{dist} = M_{calc} / (1.1 M_{fast})$  where 1.1 was used to allow for better solutions in the final run. For reference the standard Qwerty layout scored 0.395 while Bi, Smith and Zhai's quasi-Qwerty keyboard scored 0.643 – confirming that their quasi-optimization process resulted in considerably less distance for a single finger to move on average.

We discuss the triple-optimized keyboards and the Pareto process in full below. However, running our Pareto optimization process resulted in over 24 000 keyboards on the final "surface". Of these, the highest scored keyboard for finger distance metric on a standard iPhone™ style layout has a distance weight of 0.908 (figure 3). Note that the top four most common bigrams ( $E\_$ ,  $\_T$ ,  $S\_$  and  $TH$ ) are neighbours with others being near neighbours.



Figure 3: Fastest iPhone layout keyboard<sup>3</sup>

The Pareto optimization process is designed to find best solutions along the Pareto front, as such it is not good at finding bad solutions as poorer ones are discarded in favour of all-round better ones. However, it is worth contrasting the best solution found with the worst recorded at the end of the search. The poorest performing keyboard on the front for finger travel distance had a weight of 0.256 (figure 4). Tapping out a common phrase with these two keyboards casually confirms that the finger moves considerably less with the best rather than the worst keyboard.

Figure 5 compares the finger travel metric for our fastest keyboard (Fig. 3) with the standard Qwerty and Quasi-Qwerty keyboards (Fig. 2).

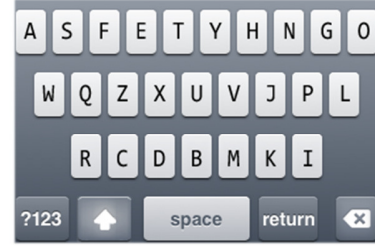


Figure 4: Slowest keyboard on final Pareto front

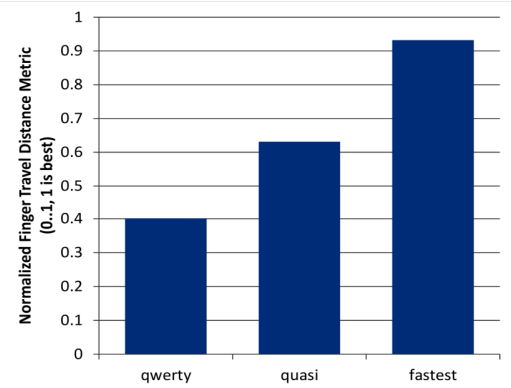


Figure 5: Finger travel distance metric comparison

#### Tap interpretation clarity metric (Neighbour Ambiguity)

Traditionally text entry methods can be categorized as *unambiguous*, where each key unambiguously maps to a character (e.g. laptop Qwerty keyboards), or *ambiguous*, where multiple characters are mapped to each key (e.g. the traditional 12-key phone pad). With an ambiguous keyboard the most common method of automatic disambiguation is to use a large dictionary (e.g. T9 [11] and [7]). Dictionary disambiguation offers the most common word in the language when a user types a key sequence, e.g. on a 12-key phone *hello* will be offered for 43556 as the most likely word given the keys *GHI DEF JKL JKL MNO*. Overall this works surprising well, with success rates estimated at around 95% ([10]). However, it does not cope with key combinations where two or more words are widely used, e.g. *home/good* and *he/if* are common examples that share the same keystrokes on a traditional phone. More complex approaches to disambiguation, e.g. [8, 12], attempted to solve this using more contextual knowledge. Alternatively, Gong and Tarasewich [10] investigated the best layout of miniature keypads to reduce the ambiguity of the keyboard layout itself by separating combinations that lead to multiple popular words. The best solution, of course, is a combination of both: a powerful contextual engine with an optimized layout to reduce the effort required by the context engine.

Modern powerful smart-phones and laptop/desktop spell correctors have blurred the distinction between ambiguous and unambiguous keyboards – they typically give users the impression of an unambiguous Qwerty layout but use increasingly complex automatic error correction algorithms

<sup>TM</sup> iPhone is a trademark of Apple Inc.

<sup>3</sup> Here we refer to iPhone-layout as a standard Qwerty key layout with 10 keys on top row, 9 middle and 7 bottom with the same tall key aspect-ratio as portrait iPhones.

to soften the solidity of the one-char-per-key rule (e.g. [4, 16, 18]). For example, typing *typung* in most desktop word processors and most touch phones will result in the word *typing* being inserted even though the user tapped the unambiguous *u* as the fourth key. Error correction has been shown to be particularly important on touch screens with small keys [18] and is seen as one of the challenges for intelligent text entry [17]. Furthermore, Allen et al. showed that, while expert touch-screen users and expert physical-keyboard users achieved roughly the same speed, both groups had higher error rates on iPhones than on mini-physical keyboard phones [1]. This implies that, although automatic error correction has come far there are still considerable problems with error correction on touch-screen mobiles.

In developing our keyboard layout one factor we wished to take into account was interpretation clarity for taps. We created a table of *bad-bigrams*, or *badgrams* for short, of keys that were ambiguous given their neighbours. This table is similar to the table used above for keyboard distance but is based on the likelihood of a one letter substitution resulting in a valid word, e.g. mistyping *for* as *fir* results in a badgram for *OI* on Qwerty keyboards. We scanned all same-length words in our corpus and assigned a frequency to each badgram found based on the more common of the two words. Summed over all words on the corpus, this resulted in *AE* being the most frequent badgram with 1 227 442 weighted occurrences (i.e. having *A* and *E* as neighbours leads to many single key tap errors giving valid words: *end* instead of *and*, *ha* instead of *he*, *been* instead of *bean* etc.). As with the bigram table, we converted to probabilities by dividing the score by the total score for all combinations to give a top badgrams<sup>2</sup> of *AE*=0.017, *AO*=0.017, *EO*=0.015, *ST*=0.015, *EI*=0.013, *IO*=0.012 and *AI*=0.012.

The aim of the tap clarity optimizer was to reduce the total ambiguity for keys that were adjacent in the layout, which should maximize the effectiveness of a spell corrector to correctly interpret taps. This metric is defined as:

$$M_{tap\_clarity} = \langle \sum_{i,j \in \alpha} \begin{matrix} \text{if neighbours}_{ij}: p_{ij} \\ \text{else: } 0 \end{matrix} \rangle$$

where  $\alpha$  is the alphabet in use (here  $a...z$ ),  $P_{ij}$  is the badgram probability for letters  $i,j$  and  $neighbours_{ij}$  is true if the keys for  $i$  and  $j$  are adjacent (vertically or horizontally) on the selected keyboard, otherwise false. For Pareto optimization, this score is again normalized to approximately the range 0...1, where 1 represents the best keyboard and 0 the worst. For reference the standard Qwerty layout scored 0.559 while quasi-Qwerty scored 0.459, showing that this layout sacrificed some spell-checking clarity in making their speed gain.

Again using a standard iPhone-layout, the best found keyboard for neighbour ambiguity had a score of 0.997 (figure 6). This keyboard should be optimal for a spell

checker to correctly interpret taps as single letter tap errors will most likely not result in a valid word (or at least not a common valid word). The most common badgrams are clearly separated (e.g. *AE*, *AO*, *EO*, *ST*). Figure 7 compares the interpretation clarity metric for this keyboard with the standard Qwerty and Quasi-Qwerty keyboards.



Figure 6: Best keyboard for spelling correction

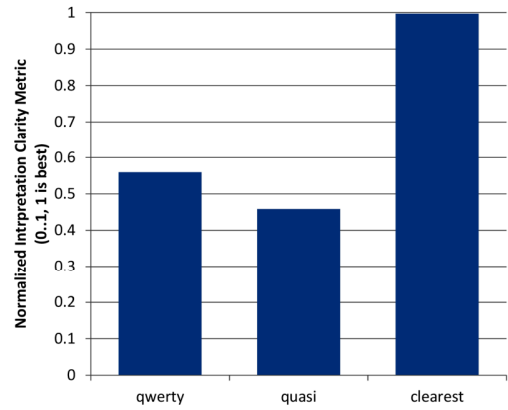


Figure 7: Comparison of Interpretation Clarity Metric

#### Familiarity to Qwerty metric

There is a long history of text entry research into alternative keyboards for touch screens. While achieving very promising expert user performance predictions, these layouts have had very low adoption rates as users tend to favour the familiar Qwerty layout. Bi, Smith and Zhai [2] proposal was a middle ground: they allowed keys to be moved around to optimize a layout but restricted the distance to 1 key away from the home key. We have followed their general approach but softened this rule by imposed a strong weighting against keys which move far from their Qwerty layout position. The effect being to allow keys more freedom but punish a keyboard design where many keys move from the Qwerty home location and severely punish keyboards where individual keys have moved far from their home location. The aim is that when users are typing with a finger on a touch screen, the keys they are aiming for will most often be in the proximity of where they expect it to be given their Qwerty experience but at the same time to give freedom for stronger optimization of other metrics.

Similarity between keyboards can be measured by scoring the distance of all keys to their home keys on a same-sized standard Qwerty keyboard. However, to increase familiarity



of the keyboard and “punish” keys that move far from their home we experimented with different familiarity metrics based on squaring, cubing and exponential function of the Euclidean distance for each key. With experimentation, squaring the distance gave the best balance between allowing movement and keeping keys near their home locations. This function gives a distance score of 0 for a key that is in the same location as on the Qwerty layout, 1 for a key that moves to its neighbour (horizontally and a key's aspect ratio vertically, e.g. 1.7 for an iPhone), and 9 for a letter that moves three keys horizontally (given the standard layout is 10 keys wide this is a high value). However, as this metric averages the score over all keys, unlike Bi et al.'s quasi-Qwerty, it does give flexibility for individual keys to move a few keys if many of the other keys stay very close to their Qwerty location.

We calculated the familiarity metric as:

$$M_{familiarity} = \langle \sum_{i \in \alpha} distance(k_i, q_i)^2 \rangle$$

where  $\alpha$  is the alphabet in use ( $a...z$ ),  $k_i$  is the location of the centre of the key on the given keyboard,  $q_i$  is its location on a same sized standard Qwerty layout keyboard, and  $distance$  is the Euclidian distance between these points. Again the score is finally normalized to the range 0...1 for Pareto optimization, where 1 represents the best keyboard found and 0 the worst. For reference the standard Qwerty layout scores 1.0 while quasi-Qwerty scores 0.850.

### TRIPLE-METRIC OPTIMIZATION PROCESS

In designing artefacts, we often have more than one criterion that we use to evaluate the final product. For example, a motor vehicle can be judged by its fuel efficiency, its ease of handling, the comfort of the ride and so on. Often these criteria conflict: a hard suspension may help with handling but be detrimental to passenger comfort. Multi-objective optimization algorithms [29] seek to create solutions to such problems by considering the optimization process across these potentially conflicting objectives. A simple way of addressing such problems is to create a single combined objective function, where each individual objective is a component in a weighted sum. However, the difficulty of coming up with an appropriate weighting for each part of the sum and the fact that this method only returns a single solution means that this is not generally the method of choice [6]. Instead, what is needed is a method which can return multiple solutions where each solution has something about it which makes it better than other solutions according to at least one of the criteria. This leads to a need to explore solutions that are Pareto optimal. If there are 3 criteria to optimize, as in this study, and we have found a Pareto optimal artefact which has the evaluation  $[x,y,z]$ , then this means there is no point in the solution set for which all criteria are equal or better. In other words, if we want to improve the score for one of the criteria along the Pareto front, we have to compromise by lowering the score for at least one of the other metrics. A point which is not Pareto optimal is said to be *dominated* - there is a Pareto

optimal solution which is better than it in at least one dimension and no worse in the others.

The search algorithm in this work is a variant of local neighbourhood search [14] adapted for use in finding a Pareto optimal set using the above three metrics: finger travel distance, spelling interpretation clarity and Qwerty familiarity. The process starts with a randomly generated set of points that are optimized locally for different weightings of the three metrics (typically 40-50 starter keyboards are created). This initial set of keyboards is taken through 2000 iterations of improvement in which local moves are made that may, or may not, improve the solution. In each iteration each keyboard in the set has a small number of keys swapped (1 key is swapped then extra keys are swapped with a probability of 25% of continuing after each swap<sup>4</sup>); if the new keyboard is better on any metric then it is added to the set; if it is also at least as good on ALL metrics than an existing solution then it dominates the existing one, which is discarded. This leads to a *Pareto front* - a set of dominant solutions on a 3D surface.

The final Pareto front for optimizing the standard Qwerty keyboard is shown in Figure 8. This shows the trade-off between the different measures with high scores being achievable only at the expense of others. It also, reassuringly, shows a convex surface showing that compromise solutions are not, overall, poorer than single optimized solutions. This front is composed of over 24 000 individual keyboards (out of the 46.7 million candidate keyboards considered in the 2000 iteration run).

### PROPOSED IMPROVED KEYBOARD LAYOUTS

The final compromise keyboard proposal is taken to be the keyboard that achieves best on average - the centre of the Pareto surface (i.e. keyboard nearest the 45° line through the space). All metrics were scaled in advance so the best

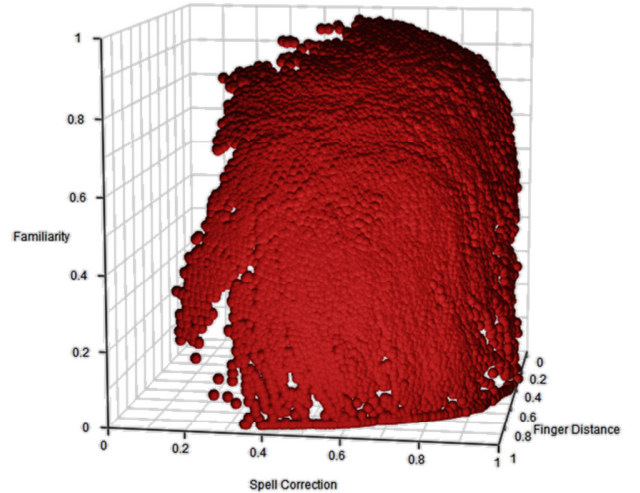


Figure 8: Pareto Front Shape

<sup>4</sup> This ensures that the Pareto curve optimization is, in a sense, complete as all combinations are reachable from any given initial keyboard layout.

(individual score) lies around 1 and the worst around 0 – ensuring the 45° point is a fair balance of the three metrics. This was achieved through iterative running of the Pareto optimization process. A small imbalance at this stage would result in us picking a different near-central solution. However, the solution space around the centre 45° selected keyboard was stable with only small changes being seen on solutions near the central one and a fairly smooth front shape near the centre (see Fig. 8). While varying per starter keyboard, most Pareto optimizations didn't change the suggested keyboard for the last 500+ iterations of 2000 optimization iterations, giving further confidence in stability of the solutions discussed below.

#### A standard Qwerty layout triple-optimized keyboard: The Sath-Trapezoidal keyboard

Using iPhone key shape (a key aspect ratio of 1.7) and a standard Qwerty layout as a starter keyboard, our triple-optimization process created the keyboard shown in Figure 9 with a score of approx. 0.69 for each metric. We will refer to this as the Sath-trapezoidal keyboard.



Figure 9: Triple optimized standard iPhone style keyboard

Table 1 summarizes the metric scores for this keyboard compared to the standard Qwerty and the Quasi-Qwerty. Overall our alternative layout achieves a considerably better finger travel distance than Qwerty and noticeably better than quasi-Qwerty. It also achieves considerably better interpretation clarity than both, but at a reduction in familiarity.

	Finger Distance	Interpretation Clarity	Qwerty Familiarity	Average score
Sath	0.694	0.695	0.694	0.694
Qwerty	0.395	0.559	1.000	0.651
Quasi-Qwerty	0.643	0.459	0.829	0.644

Table 1: Standard keyboard metrics

Given the balance of metrics and Pareto optimization process, we claim that the Sath keyboard presented here provides the best compromise between typing speed, tap interpretation for spell correction and familiarity with Qwerty to support pickup-and-use usability.

#### More rectangular layout: Sath-Rectangular

In the discussion so far we have focused on an "iPhone like Qwerty keyboard layout". This layout is a standard Qwerty

layout from typewriter and computer use but with a higher aspect ratio – approximately 1.7 for the iPhone, when measured to include surrounding grey-space, and a slightly taller 1.75 for an HTC Hero (a relatively small Android phone). The standard Qwerty layout has a trapezoidal shape, if drawn symmetrically, with 10 keys on the top row, 9 in the middle and only 7 on the bottom row (a 10-9-7 format). Full size keyboards pad the lower rows with non-alphabetic and functional keys but there are often fewer such keys on mobiles with additional characters being entered through a secondary mode. Above we presented our results for optimization using this standard trapezoidal layout and aspect ratio.

MacKenzie states that when measuring Fitts' law distance, the size of a key should be the minimum of height and width [22]. As such these tall, thin keys have effectively the same Fitts' law functions as if they were just as high as their width but with further distances between the keys vertically. As discussed above, the small keys also tend to lead to many typing errors as the key centres are very close together – for example keys of the size found on portrait mode iPhones have been shown to be significantly slower and more error prone than larger keys [18, 19]. As such we attempted to reduce the aspect ratio of keys to make them squarer, while maintaining their height and familiarity with the original Qwerty layout. We investigated Pareto optimization starting with a more rectangular 9-9-8 profile keyboard that results in a less-tall aspect ratio of approximately 1.5 for the same screen area.

Here we started our optimization process with a Qwerty layout in which the *Q* and *A* were shifted one row down to give the starter layout *WERTYUIOP QSDFGHJKL AZXCVBNM* which has a 9-9-8 profile and a familiarity score of 0.951. Using this keyboard layout and a 1.5 aspect ratio gave an improvement over the standard 10-9-7 layout with the keyboard shown in figure 10 rating approx. 0.75 for each metric.

While a relatively small numerical improvement, the buttons in this layout also have a larger hit area which should improve typing speed and reduce miss-strikes further improving spelling performance. Using the same area as an iPhone keyboard, this layout increases the key width from 4.6 to 5.2 mm – a considerable improvement of 11% in "target size" used in Fitts' law calculations. As key

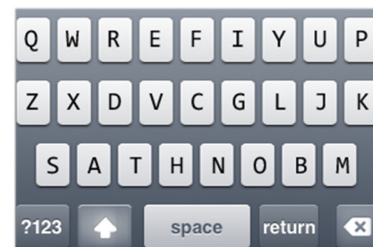


Figure 10: Optimized more rectangular keyboard layout

sizes on portrait touch-screen phones are well below research recommendations for touch screen key sizes (e.g. [26]) and have been shown to be considerably poorer than the larger keys used in landscape mode [19], this small difference may have a very significant impact on speed.

	Finger Distance	Neighbour Ambiguity	Qwerty Familiarity	Average score
Sath Trapz	0.694	0.695	0.694	0.694
Sath Rect	0.751	0.751	0.751	0.751

**Table 2: Comparing standard and rectangular layouts**

### FITTS' LAW SPEED CALCULATIONS

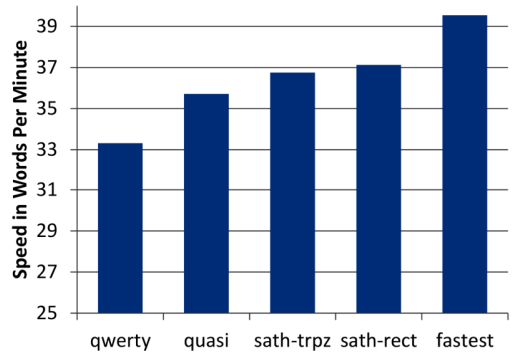
The finger distance metric used above is suitable for optimization a fixed format keyboard but cannot be used to predict text entry speed. Fitts' law [9] (Equation 1) can be used to estimate the potential speed of a keyboard layout for error-free expert text entry (e.g. [28]). As such it is worth discussing here as it gives a more concrete comparison to other keyboards through use of words-per-minute estimates. Equation 2 shows the Fitts' law calculation for weighted average time to press a key. The time to press a key is logarithmically proportional to the distance to that key while logarithmically inversely proportional to width of the target key (big keys close to the starting point are fastest targets to hit). The constants  $a$  and  $b$  have to be derived experimentally for a given device, for comparison with work of others we used the figures  $a=0.083$  and  $b=0.127$  [2, 32] in our studies despite their being derived for stylus-based keyboarding. To calculate the average time per keystroke, a weighted average is used based on the probability of bigrams in the language, so that key combinations that are struck more commonly (e.g. moving from *e* to *space*) have a proportionally higher impact on the average than rarely used key combinations (e.g. moving from *z* to *x*). The same bigram data as used for the finger travel optimizations were used here but these were compared with the table used in [25] and found to result in very small differences in predicted times.

$$\bar{T} = \sum_{i \in \alpha} \sum_{j \in \alpha} p_{ij} \left( a + b \log_2 \left( \frac{D_{ij}}{w_j} + 1 \right) \right) \quad (2)$$

For the standard Qwerty keyboard (10-9-7) we estimated an average key tap time of 0.360 s given an aspect ratio of 1.7 and the constants  $a$  and  $b$  from above. This is equivalent to a predicted expert typing speed of 33.3 words-per-minute (wpm). Bi, Smith and Zhai used the same Fitts' constants to estimate 181.2 characters-per-minute, or 36.2 wpm, for a standard Qwerty keyboard – slightly faster than our estimate. This is predominantly due to the aspect ratio of keys – Bi, Smith and Zhai followed MacKenzie's early lead in modelling touch screen entry with square keys similar to laptop keys and not the highly stretched keys now used on

touch screen phones. While the true values of  $a$  and  $b$  for finger tapping on keys below 5 mm requires to be calculated experimentally, our estimate is, we believe, unlikely to change the ordering of keyboards but will affect predicted speeds as the values of  $a$  and  $b$  used are based on studies with approximately 10 mm wide keys.

Figure 11 shows the words-per-minute estimates for our two keyboards compared with the traditional Qwerty and quasi-Qwerty (both using 1.7 aspect ratios as this matches the keyboard area of the iPhone) and, for comparison, the fastest single optimized keyboard layout we identified. This shows a predicted improvement of 10% and 11% respectively for our trapezoidal and rectangular keyboard layouts over standard Qwerty and smaller 3% and 4% predicted improvement over the quasi-Qwerty keyboard.



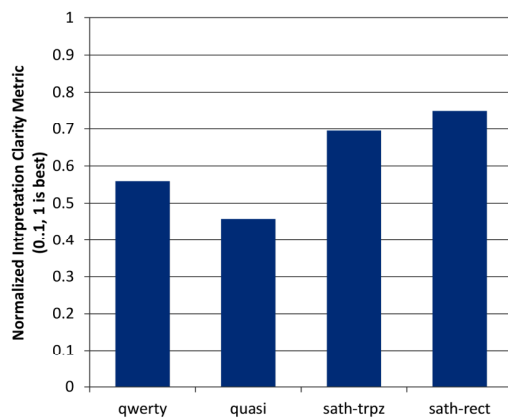
**Figure 11: Comparison of keyboard typing speeds**

However, as discussed above this does not fully take into account the increased key size with the rectangular keyboard (only the smaller vertical aspect ratio) nor does it take into account the improvement in error correction likely in practice given the larger keys. Fig. 12 shows that the two optimized keyboards presented here also have considerably better tap interpretation clarity that should lead to faster text entry as users will learn that they need to be less accurate on typing and still achieve corrected-error-free entry.

### PAPER PROTOTYPE

To investigate the initial pick-up-and-use aspects of the new keyboard we created paper prototypes of the new keyboard layout using a slightly earlier version of our optimized rectangular keyboard. These paper prototypes were correct in size and aspect ratio for an HTC Desire and were trialled with 12 students. These users were encouragingly positive and stated that they would use the keyboard when available. The students stated that they generally found keys quickly in practice typing (though the A was commented on as being moved quite far). One user commented that even for two-thumb use it felt easier as common keys were more central to the keyboard, an unintentional consequence of finger distance metric and the central space key.





**Figure 12: Comparison of keyboard interpretation clarity**

### INITIAL USER STUDIES

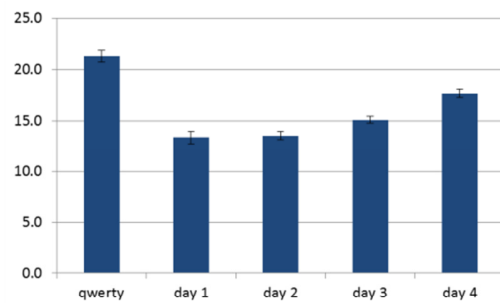
Encouraged by the paper prototype results we developed an Android implementation and ran a four day user trial with 10 regular touchscreen phones users (8 male, 2 female, mostly between 18 and 35 years old with one 36+ user) to measure their performance with rectangular-Sath over the initial learning period. Sessions lasted under 45 minutes per day in a quiet environment with subjects seated in a comfortable chair without the use of a desk.

### Procedure & Equipment

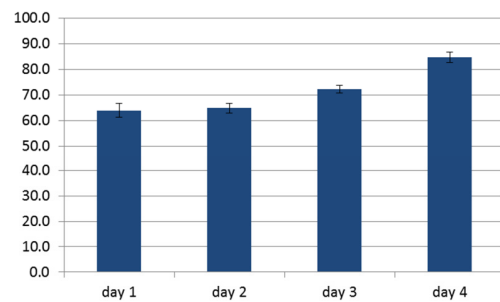
Users came at the same time for four days and were asked to enter two initial warm up phrases then 17 phrases selected randomly from MacKenzie and Soukoreff [23] standard set. There were 4 task sets (68 phrases total), randomly allocated to each participant (balanced on first day with unused phrase sets per person randomly allocated on days 2-4). To assess Qwerty performance users entered some phrases using the standard Qwerty layout (first part of day 1 and second part of day 2), all other phrases were entered using rectangular-Sath (figure 1).

Phrases were presented in the web browser of an HTC Desire S and the users typed answers into a text box on the same web page before hitting “next” to move on to the next phrase. Timing information was recorded using JavaScript based on the time from first to last key press. In line with other studies, users were asked to type as quickly as possible but accurately and were allowed to use backspace to correct mistakes they spotted “immediately” but were told not to correct mistakes they noticed later and were prevented from using editing controls except backspace.

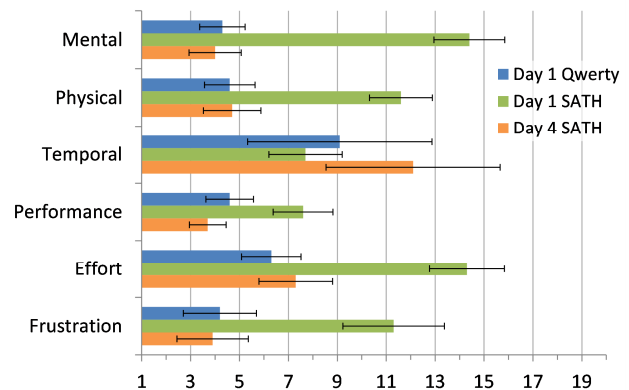
The implementation used a basic spell checking algorithm with the standard Android suggestion bar to show suggested words and highlight auto-corrections. A typed word was auto-corrected if it was not in the dictionary but a same-length dictionary word existed that was very close to the tapped locations (i.e. one tapped character was out by one key). We restricted to same length corrections to target miss-taps and not wider omitted taps, double taps or true spelling errors. The Qwerty and Sath keyboards used the same underlying code and spell corrector.



**Figure 13: Words per minute speeds**



**Figure 14: Percentage speed compared to Qwerty**



**Figure 15: NASA TLX Scores**

### Speed results

Our users averaged 21.3 wpm (stdev 7.3) using Qwerty. Their performance dropped to 13.4wpm (6.1) when using Sath for the first time but recovered to 17.7 wpm (5.2) by the fourth day of the test (figure 13 shows the daily results – Sath speed based on Sath phrases per day with Qwerty based on all Qwerty phrases as there was no significant difference in Qwerty speed between day 1 and 2).

We also analysed speed as a percentage of the users individual Qwerty performance. This analysis shows that users dropped to 64% of their individual Qwerty speed for the first block of phrases using Sath but that this recovered to 85% on the fourth day (fig 14). For comparison average Quasi-Qwerty performance was approx. 65% of average Qwerty in word-by-word tests, while their freely optimised keyboard achieved only 45% in initial use.

### Other Results

Uncorrected error rates were low throughout the study. Overall 7.9% of phrases contained a single erroneous word, with none having multiple errors. With an average phrase length of 5.6 standard words ( $\approx 5$  chars as used in wpm calculations), this equates to an error once per 71 words. On Qwerty tests, 5.3% of phrases were erroneous with a higher 8.8% of Sath phrases being erroneous (with no clear pattern over the four days). Errors from key positioning changes should result in same length typing errors. For Qwerty we found 3% (5 of 170) of phrases were correct-length but erroneous compared with 4% (19 of 510) of Sath phrases. All Qwerty errors were independently categorised as typos while 4 other errors were recorded with Sath – if these were excluded then Sath would have the same typo error rate as Qwerty in this initial use study (other errors were transposition of letters, spelling errors and typing the wrong, but semantically sensible, word).

NASA TLX forms were completed after each session (each block on days 1 and 2). These showed significantly higher workload for mental ( $p < 0.001$ , t-test,  $n = 10$ ), physical ( $p < 0.001$ ), effort ( $p < 0.001$ ), and frustration ( $p < 0.05$ ) for the new keyboard on the first day of use. However, there were no significant differences between first day Qwerty and fourth day Sath indicating that users had reduced to their Qwerty level of effort (figure 15).

At the end of the study, users were asked “if it was proven faster and less prone to spelling errors”, would they adopt this keyboard. Eight of the ten users replied positively on a 7-point scale with a mean response of +1.6 (see fig. 16).



Figure 16: Adoption preference

Several users commented that it would take some time to get up to full speed on the new layout while a couple commented that they had already got used to the new layout. A couple of comments showed some users understood the design, e.g. “I liked how letters which are close to each other in a word were close on the keyboard”.

### Study Discussion

We observed an initial performance of 64% which, after only four short sessions, had recovered to 85% of their own Qwerty performance with users typing at nearly 18 wpm in their fourth session. While not yet outperforming Qwerty they were typing at a good speed and showed good signs of continual improvement. Given the growth shown on figures 13/14 we are confident that this study shows (a) initial use is not too painful and (b) users would most likely exceed Qwerty speed within a short period of more intensive use.

### FUTURE WORK

We are currently improving the Android app to commercial quality to allow non-study usage. This will enable longitudinal trials after which we should be able to better estimate practical expert text entry speed. The constants  $a$

and  $b$  in our Fitts's law modelling are based on the best available estimates – these studies will also allow us to accurately model these for modern touch screen phones.

The Pareto curve provides a 3D surface on which all points are, in some sense, optimal given different bias on the underlying metrics. As users become familiar with the revised keyboards shown in this paper it may be possible to dynamically move forward along the front towards the origin of familiarity by building on the user's new familiarity with our keyboard. Longer trials are planned to see if users can handle a keyboard design that changes dynamically over time but in a “familiar way”. We will also investigate how this impacts on their use of the standard Qwerty, for example when using a friend's keyboard or swapping to hard keyboard phone or laptop.

We picked three metrics of speed, interpretation clarity and Qwerty-familiarity as we feel any optimization should take at least these three aspects into account. Other metrics could be included as Pareto optimization is open to any number of dimensions. For example, the language models underlying our optimization are relatively simple and do not require particularly detailed corpora so it would be worth exploring optimized keyboards for other languages (another of Kristensson's challenges [17]) or even multi-lingual optimization (c.f. [3]). We are also working on optimizing for two thumbs using more detailed timing models [21].

### CONCLUSION

This paper has introduced a new approach to keyboard optimization. We use Pareto Front optimization to optimize on three metrics in parallel: finger travel distance (speed of entry), tap interpretation clarity for spell correction (itself a new metric) and familiarity with standard Qwerty. Using our metrics we proposed two new Sath-keyboards that give a considerable improvement in finger travel distance by rearranging the keys on the standard layout keyboard and by also making the key layout more rectangular. In addition to the predicted improvement in speed we saw a considerable reduction in neighbour ambiguity that should lead to improved tap interpretation and spell correction. Fitts' law modelling confirmed a conservative improvement of 10-11% in terms of words-per-minute. When compared with Bi, Smith and Zhai's quasi-optimized keyboard [2] we show a small improvement in speed with a considerable improvement in the tap interpretation metric (but at a cost in familiarity). In user trials, users performed at 64% of their Qwerty speed on first use but this improved to 85% within four short trial sessions and showed strong signs of continued improvement. Moreover, the combined effect of less distance in typing and higher tap interpretation clarity should, in medium term use, see cumulative gains as users learn they can be less accurate with taps and achieve the same quality input. User studies are planned to accurately model finger-based entry on touch screens of these sizes and to study the impact of our improved layout and spell correction ability on input speeds over long term studies.

## ACKNOWLEDGEMENTS

Our thanks to our pre-test and main-test subjects, Tony Fowlie who developed a preliminary keyboard as part of his honours project, Scott MacKenzie for clarifications on bigram tables and to our anonymous reviewers.

## REFERENCES

1. Allen, J. M., McFarlin, L. A. and Green, T. An In-Depth Look into the Text Entry User Experience on the iPhone. In *Proc. 52nd HFES* (2008), 508-512.
2. Bi, X., Smith, B. A. and Zhai, S. Quasi-qwerty soft keyboard optimization. In *Proc. CHI 2010*, ACM Press (2010), 283-286.
3. Bi, X., Smith, B. A. and Zhai, S. Multilingual Touchscreen Keyboard Design and Optimization. *Human-Computer Interaction* (to appear 2011).
4. Clawson, J., Lyons, K., Rudnick, A., Iannucci, R. A. J. and Starner, T. Automatic whiteout++: correcting mini-QWERTY typing errors using keypress timing. In *Proc. CHI 2008*, ACM Press (2008), 573-582.
5. David, P. A. Clio and the Economics of QWERTY. *American Economic Review*, 75, 2 (1985), 332-337.
6. Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2002.
7. Dunlop, M. D. and Crossan, A. Predictive text entry methods for mobile phones. *Personal Technologies*, 4, 2 (2000).
8. Dunlop, M. D., Glen, A., Motaparti, S. and Patel, S. AdapTex: contextually adaptive text entry for mobiles. In *Proc. MobileHCI '06*, ACM Press (2006).
9. Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *J. Experimental Psychology*, 47, 6 (1954), 381-391.
10. Gong, J. and Tarasewich, P. Alphabetically constrained keypad designs for text entry on mobile devices. In *Proc. CHI '05*, ACM Press (2005), 211-220.
11. Grover, D. L., King, M. T. and Kushler, C. A. *Reduced keyboard disambiguating computer* Tegic Communications, Inc., Patent US5818437 (1998).
12. Hasselgren, J., Montnemery, E., Nuges, P. and Svensson, M. HMS: A Predictive Text Entry Method Using Bigrams. In *Proc. Workshop on Language Modeling for Text Entry Methods at EACL* (2003), 43-49.
13. Hoggan, E., Brewster, S. A. and Johnston, J. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proc. CHI '08*, ACM Press (2008), 1573-1582.
14. Hoos, H. H. and Stutzle, T. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.
15. Jones, M. P. and Martin, J. H. Contextual spelling correction using latent semantic analysis. In *Proc. ANLP 1997*, Association for Computational Linguistics (1997), 166-173.
16. Kristensson, P.-O. and Zhai, S. Relaxing stylus typing precision by geometric pattern matching. In *Proc. IUI 2005*, ACM Press (2005), 151-158.
17. Kristensson, P. Five Challenges for Intelligent Text Entry Methods. *AI Magazine*, 30, 4 (2009), 85-94.
18. Kwon, S., Lee, D. and Chung, M. K. Effect of key size and activation area on the performance of a regional error correction method in a touch-screen QWERTY keyboard. *International Journal of Industrial Ergonomics*, 39, 5 (2009), 888-893.
19. Lee, S. and Zhai, S. The performance of touch screen soft buttons. In *Proc. CHI 2009*, ACM Press (2009), 309-318.
20. Lewis, J. R., Kennedy, P. J. and LaLomia, M. J. Development of a Digram-Based Typing Key Layout for Single-Finger/Stylus Input. In *Proc. HFES* (1999).
21. MacKenzie, I. S. and Soukoreff, R. W. A model of two-thumb text entry. In *Proc. Graphics Interface 2002*, Canadian Information Processing Society (2002).
22. MacKenzie, I. S. and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17 (2002), 147-198.
23. MacKenzie, I. S. and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. In *Proc. CHI 2003*, ACM Press (2003), 754-755.
24. MacKenzie, I. S. and Zhang, S. X. The design and evaluation of a high-performance soft keyboard. In *Proc. CHI'99*, ACM Press (1999), 25-31.
25. MacKenzie, I. S., Zhang, S. X. and Soukoreff, R. W. Text entry using soft keyboards. *Behaviour & Information Technology*, 18, 4 (1999), 235-244.
26. Parhi, P., Karlson, A. K. and Bederson, B. B. Target size study for one-handed thumb use on small touchscreen devices. In *Proc. MobileHCI 2006*, ACM Press (2006), 203-210.
27. Silfverberg, M., MacKenzie, I. S. and Korhonen, P. Predicting text entry speed on mobile phones. In *Proc. CHI'00*, ACM Press (2000).
28. Soukoreff, R. W. and MacKenzie, I. S. Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour & Information Technology*, 14 (1995), 370-379.
29. Steuer, R. E. *Multiple Criteria Optimization: Theory, Computations, and Application*. John Wiley & Sons, New York, 1986.
30. Zhai, S., Hunter, M. and Smith, B. Performance Optimization of Virtual Keyboards. *Human-Computer Interaction*, 17, 2 (2002), 229-269.
31. Zhai, S., Hunter, M. and Smith, B. A. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proc. UIST'00*, ACM Press (2000), 119-128.
32. Zhai, S., Sue, A. and Accot, J. Movement model, hits distribution and learning in virtual keyboarding. In *Proc. CHI'02*, ACM Press (2002), 17-24.